

The `phylo4` S4 classes and methods

Peter Cowan & Ben Bolker

December 27, 2007

Contents

1	Introduction	1
2	Package Overview	2
3	Trees without data	2
4	Trees with data	4
5	Subsetting	6
6	Tree-walking	7
7	multiPhylo classes	7
A	Definitions/slots	7
A.1	phylo4	7
A.2	phylo4d	7
A.3	multiPhylo4	7
B	Validity checking	7
C	Hacks/backward compatibility	8

1 Introduction

This document describes the new `phylo4` S4 classes and methods, which are intended to provide a unifying standard for phylogenetic data in R. The base `phylo4` class is modeled on the the `phylo` class in `ape`. `phylo4d` and `multiPhylo4` extend the `phylo4` class to include data or multiple tree respectively. In addition to describing the classes and methods this vignette gives examples of how they might be used.

2 Package Overview

The phylobase package currently implements functions and data structures that implement the following:

- Data structures for storing a tree and multiple trees
- A data structure for storing a tree with associated tip and node data
- A data structure for storing multiple trees with one set of tip data
- Functions for reading nexus files into the above data structures
- Functions for converting between the above data structures and `ape` `phylo` objects as well as `ade4` `phylog` objects
- Functions for subsetting, replacing, and plotting the above structures

3 Trees without data

You can start with a tree — an object of class `phylo` from the `ape` package (e.g., read in using the `read.tree()` or `read.nexus()` functions), and convert it to a `phylo4` object.

For example, get the *Geospiza* data from the `geiger` package:

```
> library(geiger)
> data(geospiza)
> names(geospiza)

[1] "geospiza.tree" "geospiza.data"
```

Convert the S3 tree to a S4 `phylo4` object using the `as()` function:

```
> library(phylobase)
> g1 <- as(geospiza$geospiza.tree, "phylo4")
> g1
```

Phylogenetic tree with 14 tips and 13 internal nodes

Tip labels:

fuliginosa, fortis, magnirostris, conirostris, scandens, difficilis, ...

Node labels:

N01, N02, N03, N04, N05, N06, ...

Edge labels:

E16, E17, E18, E19, E20, E21, ...

Rooted; includes branch lengths

Note that the nodes and edges are given default names if the tree contains no node or edge names.

The `summary` method gives a little extra information, including information on branch lengths:

```
> summary(g1)
```

No root edge.

Phylogenetic tree : g1

Number of tips : 14

Number of nodes : 13

Branch lengths:

mean : 0.1764008

variance : 0.04624379

distribution :

	Min.	1st Qu.	Median	3rd Qu.	Max.
	0.00917	0.04985	0.08000	0.21910	0.88080

Print tip labels:

```
> labels(g1)
```

[1]	"fuliginosa"	"fortis"	"magnirostris"	"conirostris"	"scandens"
[6]	"difficilis"	"pallida"	"parvulus"	"psittacula"	"pauper"
[11]	"Platyspiza"	"fusca"	"Pinaroloxias"	"olivacea"	

Print internal node labels (R automatically assigns values):

```
> NodeLabels(g1)
```

[1]	"N01"	"N02"	"N03"	"N04"	"N05"	"N06"	"N07"	"N08"	"N09"	"N10"	"N11"	"N12"
[13]	"N13"											

Print edge labels (also automatically assigned):

```
> EdgeLabels(g1)
```

[1]	"E16"	"E17"	"E18"	"E19"	"E20"	"E21"	"E22"	"E23"	"E24"	"E1"	"E2"	"E3"
[13]	"E4"	"E5"	"E6"	"E25"	"E7"	"E26"	"E27"	"E8"	"E9"	"E10"	"E11"	"E12"
[25]	"E13"	"E14"										

Is it rooted?

```
> isRooted(g1)
```

[1] TRUE

Which node is the root?

```
> rootNode(g1)
```

```
[1] 15
```

Does it have any polytomies?

```
> hasPoly(g1)
```

```
[1] FALSE
```

Does it have branch lengths?

```
> hasEdgeLength(g1)
```

```
[1] TRUE
```

You can modify labels and other aspects of the tree — for example,

```
> labels(g1) <- tolower(labels(g1))
```

You can use `plot(g1)` to plot the tree (at this point, this just calls the `plot.phylo` command from the `ape` package).

Q: what else do people want to do to check their trees?

4 Trees with data

The `phylo4d` class matches trees with data. (**fixme: need to be able to use `ioNCL!`**) or combine it with a data frame to make a `phylo4d` (tree-with-data) object.

Now we'll take the *Geospiza* data from `geospiza$geospiza.data` and merge it with the tree. However, since *G. olivacea* is included in the tree but not in the data set, we will initially run into some trouble:

```
> g2 <- phylo4d(g1, geospiza$geospiza.data)
```

gives

```
Error in check_data(res, ...) :
```

```
Tip data names are a subset of tree tip labels  
(missing data names: platypiza,pinaroloxias,olivacea)  
(extra data names: Pinaroloxias,Platypiza)
```

We have two problems — the first is that we forgot to lowercase the labels on the data to match the tip labels:

```
> gdata <- geospiza$geospiza.data  
> row.names(gdata) <- tolower(row.names(gdata))
```

To deal with the second problem (missing data for *G. olivacea*), we have a few choices. The easiest is to use `missing.tip.data="OK"` to allow R to create the new object:

```
> g2 <- phylo4d(g1, gdata, missing.tip.data = "OK")
```

(setting `missing.tip.data` to "warn" would create the new object but print a warning).

Another way to deal with this would be to use `prune()` to drop the offending tip from the tree first:

```
> g1B <- prune(g1, "olivacea")
> phylo4d(g1B, gdata)
```

You can summarize the new object:

```
> summary(g2)
```

No root edge.

Phylogenetic tree : `as(object, "phylo4")`

```
Number of tips      : 14
Number of nodes     : 13
Branch lengths:
  mean              : 0.1764008
  variance          : 0.04624379
  distribution :
    Min. 1st Qu. Median 3rd Qu.    Max.
0.00917 0.04985 0.08000 0.21910 0.88080
```

Comparative data:

Tips: data.frame with 14 taxa and 5 variables

wingL	tarsusL	culmenL	beakD
Min. :3.975	Min. :2.807	Min. :1.974	Min. :1.191
1st Qu.:4.189	1st Qu.:2.929	1st Qu.:2.187	1st Qu.:1.941
Median :4.235	Median :2.980	Median :2.311	Median :2.073
Mean :4.236	Mean :2.991	Mean :2.333	Mean :2.083
3rd Qu.:4.265	3rd Qu.:3.039	3rd Qu.:2.430	3rd Qu.:2.347
Max. :4.420	Max. :3.271	Max. :2.725	Max. :2.824
NA's :1.000	NA's :1.000	NA's :1.000	NA's :1.000

gonysW
Min. :1.401
1st Qu.:1.845
Median :1.962
Mean :2.014
3rd Qu.:2.222
Max. :2.676
NA's :1.000

Object contains no node data.

Or use `tdata()` to extract the data (i.e., `tdata(g2)`). By default, `tdata()` will retrieve tip data, but you can also get internal node data only (`tdata(tree,"node")`) or — if the tip and node data have the same format — all the data combined (`tdata(tree,"allnode")`).

Plotting calls `plot.phylog` from the `ade4` package.

5 Subsetting

The `subset` command offers a variety of ways of extracting portions of a `phylo4` or `phylo4d` tree, keeping any tip/node data consistent.

tips.include give a vector of tips (names or numbers) to retain

tips.exclude give a vector of tips (names or numbers) to drop

mrca give a vector of node or tip names or numbers; extract the clade containing these taxa

node.subtree give a node (name or number); extract the subtree starting from this node

Different ways to extract the *fuliginosa-scandens* clade:

```
> subset(g2, tips.include = c("fuliginosa", "fortis", "magnirostris",  
+ "conirostris", "scandens"))  
> subset(g2, node.subtree = "N07")  
> subset(g2, mrca = c("scandens", "fortis"))
```

One could drop the clade by doing

```
> subset(g2, tips.exclude = c("fuliginosa", "fortis", "magnirostris",  
+ "conirostris", "scandens"))  
> subset(g2, tips.exclude = allDescend(g2, MRCA(g2, c("difficilis",  
+ "fortis"))))
```

Another approach is to pick the subtree graphically, by plotting the tree and using `identify`, which returns the identify of the node you click on with the mouse.

```
> plot(g1)  
> n1 <- identify(g1)  
> subset(g2, node.subtree = n1)
```

6 Tree-walking

`getDescend`, `getAncest`, `allDescend`, `allAncest`, `getNodeByLabel`, `getLabelByNode`: not sure about the names or functionality of these (how much do we work in terms of labels and how much in terms of internal numbers?)

7 multiPhylo classes

A Definitions/slots

A.1 phylo4

Like `phylo`, the main components of the `phylo4` class are:

edge an $N \times 2$ matrix of integers, where the first column ...

edge.length numeric list of edge lengths (length N or empty)

Nnode integer, number of nodes

tip.label character vector of tip labels (required)

node.label character vector of node labels (maybe empty)

root.edge integer defining root edge (maybe NA)

We have defined basic methods for `phylo4:show`, `print` (copied from `print.phylo` in `ape`), and a variety of accessor functions (see help files). `summary` does not seem to be terribly useful in the context of a “raw” tree, because there is not much to compute: **end users?**

Print method: add information about (ultrametric, scaled, polytomies (zero-length or structural))?

A.2 phylo4d

The `phylo4d` class extends `phylo4` with data. Tip data, (internal) node data, and edge data are stored separately, but can be retrieved together or separately with `tdata(x, "tip")` or `tdata(x, "all")`.

edge data can also be included — is this useful/worth keeping?

A.3 multiphylo4

B Validity checking

- number of rows of edge matrix (N) == length of edge-length vector (if > 0)
- (number of tip labels)+(nNode)-1 == N

- data matrix must have row names
- row names must match tip labels (if not, spit out mismatches)
-

Default node labels:

C Hacks/backward compatibility

Hilmar Lapp very kindly showed a way to hack the `$` operator so that it would provide backward compatibility with code that is extracting internal elements of a `phylo4`. The basic recipe is:

```
> setMethod("$", "phylo4", function(x, name) {
+   attr(x, name)
+ })
```

but this has to be hacked slightly to intercept calls to elements that might be missing. For example, `ape` detects whether log-likelihood, root edges, node labels, etc. are missing by testing whether they are `NULL`, whereas missing items are represented in `phylo4` by zero-length vectors in the slots (or `NA` for the root edge) — so we need code like

```
> if (!hasNodeLabels(x)) NULL else x@node.label
```

to handle these cases.